

REMARKS

Applicant is in receipt of the Office Action mailed January 29, 2004. Claims 11 – 67 remain pending in the present application. Reconsideration is respectfully requested in light of the following remarks.

Double Patenting Rejection:

Claim 11 is provisional rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 11 of co-pending Application No. 09/714,050. Should this rejection become non-provisional, Applicant will consider filing a terminal disclaimer or present reasons traversing the rejection.

Section 103(a) Rejection:

The Office Action rejected claims 11-12, 14, 18-20, 23-27, 29-33, 34-36, 40-44, 46-49, 50-52, 54-56, 58-59, 61-63, 65 and 67 under 35 U.S.C. § 103(a) as being unpatentable over Wang (U.S. Patent 6,292,936) in view of Hillson et al. (U.S. Patent 6,094,644) (hereinafter “Hillson”). Claim 13 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Hillson and further in view of “The Principles of Computer Hardware, Third Edition” by Alan Clements, 2000 (hereinafter “Clements”). Claims 15-17, 37-39, 57 and 64 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Hillson and further in view of “Load-Time Structural Reflection in JAVA” by Shigeru Chiba, June 2000 (hereinafter “Chiba”). Claims 21, 22, 45, 53 60 and 66 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Hillson and further in view of “The IR to VMx86 Translation Module Specification” by Chris Lattner, Dec. 1999 (hereinafter “Lattner”).

Regarding claim 11, the Examiner asserts that Wang teaches “a first process detecting one or more script language instructions in a markup language document; ... [and] generating an intermediate representation of the one or more script language

instructions, wherein the intermediate programming language representation of the one or more script language instructions is different from the script language.” Applicant respectfully disagrees with the Examiner’s interpretation of Wang and asserts that Wang fails to teach generating a intermediate representation of the one or more script language instructions.

Wang teaches a system wherein multiple intermediate sources are created from an original source document to enable multiple processors to interpret and process their respective intermediate source. Wang further teaches that when creating the intermediate source for a specific processor, those blocks from the original source that are not appropriate for the specific processor are translated into synchronization and/or notification tokens to allow synchronization between the processors (Wang, column 3, lines 31-49). Wang does this so that each processor has an intermediate file containing only those instructions it needs to process and the synchronization information necessary for ordering the overall processing of the original source (Wang, column 1, lines 44-51).

Wang gives as an example an input source containing HTML and VisualBasic Script. In this example, the HTML processor creates two intermediate sources, one for a Java Virtual Machine, intermediate source 200, and one for a VisualBasic Script interpreter, intermediate source 202. Intermediate source 202 for the VisualBasic Script interpreter contains the original VisualBasic Script instructions that it needs to process, and synchronization tokens where the original non-VisualBasic Script instructions were in the original source (Wang, column 3, lines 44-49, and Fig. 2).

Likewise, intermediate source 200, for the JVM contains the HTML instructions for the JVM to process and notify and wait method invocations where the VisualBasic Script instructions were in the original source (Wang, column 3, lines 31 –43, and Fig. 2). Wang states this clearly with, “[t]he HTML Parser translates the remaining VisualBasic Script blocks in the original input source into notify method and wait method invocations” (Wang, column 4, line 65-column 5, line 1).

Therefore, Applicant asserts when Wang creates intermediate sources, he is not translating embedded script instructions, but merely replacing them with uniform synchronization calls or tokens. For example, every VisualBasic Script block is translated into notification and wait method invocations for the VJM, and every HTML block is translated into synchronization tokens for the VisualBasic Script interpreter. Clearly these translations are not representations of the original script instructions. Therefore, Applicant asserts that Wang fails to teach generating an intermediate representation of the one or more script language instructions.

The Examiner supports his assertion that Wang teaches the intermediate representation of the one or more script language instructions is different from the script language, by stating, “Wang teaches converting VisualBasic Script instructions into a Java thread object, which is not the script language instructions”. As shown above, Wang teaches replacing the script instructions, not with a *representation* of those instructions, but with synchronization calls. Further, these synchronizations calls do not contain any information about the original script instructions they are replacing, but merely inform one processor to signal another processor and wait for response.

Thus, Applicant asserts that the translations of Wang are not intermediate *representations* of the script language instructions. Wang therefore fails to teach that the intermediate representation of the one or more script language instructions is different from the script language.

In further regard to claim 11, the Examiner states that Wang teaches, “interpreting and executing each of the intermediate representation instructions by using program objects to implement the instructions; ... wherein said interpreting and executing produces results in accordance with the original one or more script language instructions.” The Examiner supports his contention by stating “[t]he interpreter inherently converts the script into an equivalent intermediate form.” Applicant disagrees with the Examiner’s interpretation of Wang.

As shown above, Wang fails to teach an intermediate *representation* of the script instructions because Wang replaces the script instructions with synchronization method invocations. These synchronization method invocations are the only program objects Wang teaches and since they always perform the same actions (invoking run, notify, and wait method) (Wang, column 4, lines 48-58), in every instance, they clearly do not produce results in accordance with the original one or more script language instructions as the Examiner implies.

Therefore, applicant asserts that Wang clearly fails to teach interpreting and executing each of the intermediate representation instructions by using program objects to implement the instructions, wherein said interpreting and executing produces results in accordance with the original one or more script language instructions.

Further regarding claim 11, the Examiner admits that Wang fails to teach that the first process is implemented in a platform-independent programming language, but that Hillson teaches a web browser implemented in a virtual machine. The Examiner contends that it would be obvious to combine the interpreter based embedded scripting environment of Wang with the time recording system of Hillson so that the translator of Wang “is implemented in a platform-independent programming language, as taught by Hillson, since this allows an markup language parser, such as web browser, to be platform independent, and hence much more portable.” (Page 4, lines 13-16).

Wang however, does not teach that his translator is a web browser, but rather a HTML Parser (Wang, column 2, lines 58-60) run on a server system (Wang, figure 1, item 114, and column 23 lines 1-5). Applicant can find no teaching or suggestion in Wang of the desirability or benefit of using a web browser to perform the functionality of HTML parser 114. Wang clearly teaches that the HTML parser is enabling multiple run-time processors, which are part of the Web daemon 108, “to create the Web content.” (Wang, column 2, lines 49-58). Applicant asserts that Web browsers are used to view Web content and therefore could not be used to produce such content. Further, applicant

can see no feasible way for such a web browser to be run on a server system to create web content as is Wang's HTML parser.

In light of the above remarks, Applicants assert that the rejection of claim 11 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 11 apply to claims 27, 34, 50-52, and 62.

Regarding claim 12, the Examiner states that "Wang teaches: (a) the web browser passing execution to an interpreter engine after script language detection; (b) wherein said generating, interpreting, executing, and accessing one or more program objects in performed by the interpreter engine." Applicant disagrees with the Examiner's interpretation of Wang. Wang fails to teach a web browser passing execution to an interpreter engine after script language detection. In contrast, Wang teaches using a web browser to connect to a server system that employs an HTML parser and multiple interpreters (Wang, column 2, lines 38-42, and lines 49-55). Further, Applicant can find no teaching in Hillson regarding a web browser passing execution to an interpreter engine that performs said generating, interpreting, executing, and accessing one or more program objects as Examiner contends.

In light of the above remarks, Applicants assert that the rejection of claim 12 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 12 apply to claims 35, and 63.

Regarding claim 25, the Examiner contends that Wang teaches "a Java-based web browser executing within a Java Virtual Machine that executes the HTML parser." Applicant disagrees with the Examiner's interpretation of Wang. Applicant asserts that Wang fails to teach a Java-based web browser. Further, the Examiner states, in the office action mailed January 29, 2004, "Wang does not teach that the Web browser is executed within a Java Virtual Machine" (page 4, line 22 – page 5, line 1). Also, Wang fails to teach that a web browser executes the HTML parser. In fact, Wang teaches away from

this by describing how the server system executes the HTML parser (Wang, column 3, lines 1-5) while web browsers are used from separate machines to connect to the system server (Wang, column 2, lines 38-42, and lines 49-55, and Figure 1).

In light of the above remarks, Applicants assert that the rejection of claim 25 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 25 apply to claims 33, 36, and 67.

In regard to claim 26, Applicant argued in the response to the previous office action that Wang fails to teach or suggest “a Web browser implemented in a platform-independent programming language examining a current tag of a markup language document marking the beginning of a portion of the markup language document,” or “if said examining determines the current tag of the markup language document identifies the portion of the markup language document as comprising script language instructions an interpreter engine implemented in the platform-independent programming language” as recited in Applicant’s claim 26.

In response, the Examiner claims, “the HTML parser in Wang can be seen as a Web browser (Figure 1, item 114), in that it parses and interprets HTML code, and handles script instructions.” Applicant disagrees and asserts that a Web browser is very different from a HTML parser. In Figure 1, Wang clearly shows Web browser 104 being separate and clearly distinct from HTML parser 114.

Additionally, Wang teaches that the HTML parser is enabling multiple run-time processors, which are part of the Web daemon 108, “to create the Web content.” (Wang, column 2, lines 49-58). Applicant asserts that Web browsers are used to view Web content, not create it.

Therefore, in light of the above remarks, Applicants assert that the rejection of claim 26 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 26 apply to claim 48.

Regarding claim 56, the Examiner correctly points out that in the Office Action Response dated October 14, 2003, claim 56 was erroneously listed as having been amended. Claim 56 was not amended in the previous response.

Further regarding claim 56, the Examiner contends that a combination of Wang and Hillson would produce a system including a device with a Java Virtual Machine, an interpreter engine, a Web browser, and an HTML parser that detects scripts in the HTML document, and provides the scripts to the interpreter engine where the scripts are executed, as taught by Wang, where the Web browser is executed within a Java Virtual Machine, as taught by Hillson.

Applicant asserts however, that Wang in view of Hillson fails to teach a system comprising a device; a Java Virtual Machine executable within the device; an interpreter engine executable within the device; and a Web browser executable within the Java Virtual Machine to: detect one or more script language instructions in a markup language document; and pass execution to the interpreter engine in response to said detecting; wherein the interpreter engine is executable within the device to: generate an intermediate representation of the detected one or more script language instructions; and interpret and execute each of one or more instructions of the intermediate representation to produce results in accordance with the original one or more script language instructions; wherein, in said interpreting and executing, the interpreter engine is further executable within the device to access one or more program objects to implement the one or more instructions of the intermediate representation as recited in Applicants claim 56.

Specifically, Wang in view of Hillson fails to teach or suggest a system wherein the interpreter engine is executable within the device to generate an intermediate representation of the detected one or more script language instructions; and interpret and execute each of one or more instructions of the intermediate representation to produce results in accordance with the original one or more script language instructions.

As shown above regarding claim 11, Wang fails to teach the creation of an intermediate *representation* of script language instructions. In contrast, when creating intermediate sources, Wang replaces the script instructions with synchronization method invocations (Wang, column 4, line 64 – column 5, line 7). Hillson teaches a time-recording system for recording the actual time used by a service that makes data requests (Hillson, abstract) and fails to teach or suggest anything regarding detecting or translating embedded script instructions.

Further, Wang teaches that an HTML parser detects embedded script instructions and generates intermediate sources customized with synchronization tokens to be handed off to the multiple run-time interpreters. Applicant can find no teaching in Wang or Hillson regarding an interpreter engine generating an intermediate representation of the detected one or more script language instructions.

Additionally, Applicants can find no teaching or suggestion in Wang or Hillson of the interpreter engine executable within the device to access one or more program objects to implement the one or more instructions of an intermediate representation.

Therefore, in light of the above remarks, Applicants assert that the rejection of claim 56 is not supported by the cited art and withdrawal of the rejection is respectfully requested.

Regarding claim 61, The Examiner states that the reasons and logic regarding this rejection would be found in the office action mailed on July 14, 2003. Applicant however, can find no rejection whatsoever regarding claim 61 in the earlier office action and therefore asserts that the reject of claim 61 is improper and respectfully requests its removal.

Regarding claim 13, the Examiner contends that Clements teaches using the stack structure to hold instructions that are executed by popping the instructions off of the

stack. Applicant disagrees with the Examiner's interpretation of Clements. In fact, Clements clearly fails to teach using a stack structure to hold *instructions*.

Clements teaches storing data on a stack and further teaches that the 68K family of processors includes instructions that manipulate and use the *contents* of the stack. Clements teaches, "a computer can transfer *data* between memory and the stack, and perform monadic operations on the top item of the stack, or dyadic operations on the top two items of the stack" (Emphasis added) (Clements, Section 6.5, paragraph 4). Clements further teaches the use of the stack to store the parameters and the return addresses of subroutines (Clements, Sections 6.5.2 and 6.5.4).

Applicant can find no reference in Clements regarding using a stack structure to hold instructions that are executed by popping the instructions off of the stack as the Examiner contends. Furthermore, Applicant notes that the Examiner has neglected to cite any particular passages in Clements supporting his contention.

Therefore, in light of the above remarks, Applicants assert that the rejection of claim 13 is not supported by the cited art and withdrawal of the rejection is respectfully requested.

Regarding claim 17, the Examiner takes official notice that removing methods and fields from a program object is well known, especially when methods and fields become outdated, and hence no longer have any function. However, even if removing methods from a program object may be well known *in other circumstances*, removing methods of a program object is not well known as part of generating an intermediate representation of the one or more script language instructions. It is not enough for the Examiner to state that something is well known. As the Federal Circuit stated in *In re Kotzab*, 55 USPQ2d 1313, 1316 (Fed. Cir. 2000):

Most if not all inventions arise from a combination of old elements. However, identification in the prior art of each individual part claimed is insufficient to defeat patentability of the whole claimed invention.

The Examiner has not cited any reference that suggests removing methods of a program object as part of generating an intermediate representation of the one or more script language instructions. Thus, Applicant respectfully requests the removal of the rejection of claim 17. Similar remarks as discussed above in regard to claim 17 apply to claim 39.

Further regarding claims 17 and 39, Applicant respectfully reminds the Examiner that merely stating that individual aspects of a claimed invention are well known does not render the combination well known without some objective reason to combine the individual teachings. *Ex parte Levengood*, 28 USPQ2d 1300. The Examiner did not provide any such reasoning. Therefore, the rejection is further improper. As the Court of Appeals for the Federal Circuit recently explained in *In re Sang Su Lee*, Docket No. 00-1158 (Fed. Cir. January 18, 2002), conclusory statements such as those provided by the Examiner that a claim limitation is well known or common knowledge do not fulfill the Examiner's obligation. "Deficiencies of the cited references cannot be remedied by the [Examiner's] general conclusions about what is 'basic knowledge' or 'common sense.'" *In re Zurko*, 59 USPQ2d 1693, 1697 (Fed. Cir. 2001).

Regarding claim 21, Applicant disagrees with the Examiner's assertion that "it would have been obvious ... to perform the detecting, generating, interpreting, executing, and accessing steps of claim 11, as taught by Wang and Hillson, where each of the instructions in the intermediate representation is represented as one or more Java objects, as taught by Lattner." As shown above, Wang as modified by Hillson fails to teach the method of claim 11.

Furthermore, Lattner teaches an Instruction class which is designed to exist as a node in a linked list and generate a legal 80x86 assembly language string for an intermediate language instruction associated with that particular node in the list. (page 2). The 80x86 assembly language instructions are of a wholly distinct and different character than embedded script instructions. Just because Lattner teaches the use of a Java-based Instruction class to represent 80x86 assembly instructions, does not suggest use of a Java-based Instruction class to represent script instructions. There is no suggestion in the prior

art to apply a Lattner's Java-based Instruction class to represent anything other than 80x86 assembly instructions. Assembly instructions and script instructions are handled differently in computer systems. Prior art teachings in regard to assembly instructions do not automatically extend to script instructions.

Therefore, in light of the above remarks, Applicants assert that the rejection of claim 21 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 21 apply to claims 45, 53, 60, and 66.

Applicant also asserts that numerous ones of the other dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

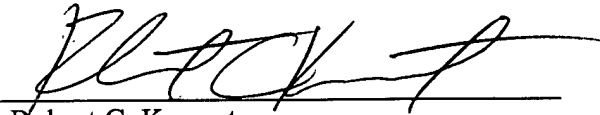
Applicant submits the application is in condition for allowance, and notice to that effect is requested.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicant hereby petitions for such extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-60100/RCK.

Also enclosed herewith are the following items:

- ☒ Return Receipt Postcard
- ☐ Notice of Change of Address
- ☐ Fee Authorization Form authorizing a deposit account debit in the amount of \$_____ for fees (_____).
- ☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 20, 2004